US-PAT-NO: 5956037
DOCUMENT-IDENTIFIER: US 5956037 A
TITLE: Video information providing/receiving system

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Image

☐ 6. Document ID: US 5943055 A
Entry 6 of 33 File: USPT Aug 24, 1999

US-PAT-NO: 5943055
DOCUMENT-IDENTIFIER: US 5943055 A
TITLE: Computer interface method and system

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Image

☐ 7. Document ID: US 5887193 A
Entry 7 of 33 File: USPT Mar 23, 1999

US-PAT-NO: 5887193
DOCUMENT-IDENTIFIER: US 5887193 A
TITLE: System for loading control information from peripheral devices which are represented as objects to a controller in a predetermined format in response to connection operation

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Image

☐ 8. Document ID: US 5877842 A
Entry 8 of 33 File: USPT Mar 2, 1999

US-PAT-NO: 5877842
DOCUMENT-IDENTIFIER: US 5877842 A
TITLE: Digital Dailies

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Image

☐ 9. Document ID: US 5852435 A
Entry 9 of 33 File: USPT Dec 22, 1998

US-PAT-NO: 5852435
DOCUMENT-IDENTIFIER: US 5852435 A
TITLE: Digital multimedia editing and data management system

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Image

☐ 10. Document ID: US 5852664 A
Entry 10 of 33 File: USPT Dec 22, 1998

US-PAT-NO: 5852664
DOCUMENT-IDENTIFIER: US 5852664 A
TITLE: Decode access control for encoded multimedia signals

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Image

Generate Collection

WEST

Help | Logout

Main Menu | Search Form | Posting Counts | Show S Numbers | Edit S Numbers

Generate Collection

**Search Results** - Record(s) 1 through 10 of 33 returned.

1. Document ID: US 6031529 A
Entry 1 of 33 File: USPT Feb 29, 2000

US-PAT-NO: 6031529
DOCUMENT-IDENTIFIER: US 6031529 A
TITLE: Graphics design software user interface

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Image

2. Document ID: US 6011895 A
Entry 2 of 33 File: USPT Jan 4, 2000

US-PAT-NO: 6011895
DOCUMENT-IDENTIFIER: US 6011895 A
TITLE: Keyword responsive variable content video program

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Image

3. Document ID: US 6002833 A
Entry 3 of 33 File: USPT Dec 14, 1999

US-PAT-NO: 6002833
DOCUMENT-IDENTIFIER: US 6002833 A
TITLE: Disc storing a variable-content-video and a user interface

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Image

4. Document ID: US 5999173 A
Entry 4 of 33 File: USPT Dec 7, 1999

US-PAT-NO: 5999173
DOCUMENT-IDENTIFIER: US 5999173 A
TITLE: Method and apparatus for video editing with video clip representations displayed along a time line

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Image

5. Document ID: US 5956037 A
Entry 5 of 33 File: USPT Sep 21, 1999

| Terms | Documents |
|---|---|
| l1 and title | 33 |

Display 10 Documents including document number 11

**Display Format:** TI Change Format

Main Menu | Search Form | Posting Counts | Show S Numbers | Edit S Numbers

Help Logout

WEST

# Freeform Search

**Database:** US Patents Full-Text Database

**Term:** l1 and title

**Display** 10 **Documents in Display Format:** TI

**Generate:** ○ Hit List ◉ Hit Count ○ Image

Search | Clear | Help | Logout

Main Menu | Show S Numbers | Edit S Numbers

## Search History

| DB Name | Query | Hit Count | Set Name |
|---|---|---|---|
| USPT | l1 and title | 33 | L4 |
| USPT | l1 and inventory and audio | 4 | L3 |
| USPT | inventory and audio | 1277 | L2 |
| USPT | edi$4 near3 play and list | 91 | L1 |

WEST

Help Logout

Main Menu | Search Form | Posting Counts | Show S Numbers | Edit S Numbers

Generate Collection

**Search Results** - Record(s) 11 through 20 of 33 returned.

☐ 11. Document ID: US 5801685 A
Entry 11 of 33 File: USPT Sep 1, 1998

US-PAT-NO: 5801685
DOCUMENT-IDENTIFIER: US 5801685 A
TITLE: Automatic editing of recorded video elements sychronized with a script text read or displayed

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Image

☐ 12. Document ID: US 5792971 A
Entry 12 of 33 File: USPT Aug 11, 1998

US-PAT-NO: 5792971
DOCUMENT-IDENTIFIER: US 5792971 A
TITLE: Method and system for editing digital audio information with music-like parameters

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Image

☐ 13. Document ID: US 5724472 A
Entry 13 of 33 File: USPT Mar 3, 1998

US-PAT-NO: 5724472
DOCUMENT-IDENTIFIER: US 5724472 A
TITLE: Content map for seamlessly skipping a retrieval of a segment of a video

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Image

☐ 14. Document ID: US 5717814 A
Entry 14 of 33 File: USPT Feb 10, 1998

US-PAT-NO: 5717814
DOCUMENT-IDENTIFIER: US 5717814 A
TITLE: Variable-content video retriever

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Image

☐ 15. Document ID: US 5696869 A
Entry 15 of 33 File: USPT Dec 9, 1997

US-PAT-NO: 5696869
DOCUMENT-IDENTIFIER: US 5696869 A
TITLE: Variable-content-video provider system

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Image

☐ 16. Document ID: US 5680533 A
Entry 16 of 33 File: USPT Oct 21, 1997

US-PAT-NO: 5680533
DOCUMENT-IDENTIFIER: US 5680533 A
TITLE: Videographics program/video game fabricating system and method

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Image

☐ 17. Document ID: US 5680534 A
Entry 17 of 33 File: USPT Oct 21, 1997

US-PAT-NO: 5680534
DOCUMENT-IDENTIFIER: US 5680534 A
TITLE: Video game/videographics program fabricating system and method with superimpose control

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Image

☐ 18. Document ID: US 5649171 A
Entry 18 of 33 File: USPT Jul 15, 1997

US-PAT-NO: 5649171
DOCUMENT-IDENTIFIER: US 5649171 A
TITLE: On-line video editing system

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Image

☐ 19. Document ID: US 5640320 A
Entry 19 of 33 File: USPT Jun 17, 1997

US-PAT-NO: 5640320
DOCUMENT-IDENTIFIER: US 5640320 A
TITLE: Method and apparatus for video editing and realtime processing

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Image

☐ 20. Document ID: US 5621877 A
Entry 20 of 33 File: USPT Apr 15, 1997

US-PAT-NO: 5621877
DOCUMENT-IDENTIFIER: US 5621877 A
TITLE: Rewinding time-based script sequences

Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC | Image

Generate Collection

| Terms | Documents |
|---|---|
| l1 and title | 33 |

Display 10 Documents including document number 21

**Display Format:** TI Change Format

Main Menu | Search Form | Posting Counts | Show S Numbers | Edit S Numbers

Help Logout

## Document Number 19

Entry 19 of 91 File: USPT Apr 6, 1999

DOCUMENT-IDENTIFIER: US 5892915 A
TITLE: System having client sending edit commands to server during transmission of continuous media from one clip in play list for editing the play list

ABPL:
A protocol and interface provides continuous play over multiple clips for extended periods of time, allows a play-list to be edited dynamically after being given to the video server and during playback of clips in the play-list, allows some notion of "current time" to be used during the streaming of continuous media data, and supports features of the "Louth Automation" video disk communications protocol. Preferably, the client application first creates a session with a play-list containing a fixed number of entries; the number should be as small as possible consistent with the client's requirements. The client edits this play-list by appending the first few clips and then starts the session playing. Each time transmission of video data of a clip is completed, the clip is removed from the head of the play-list, all other clips are moved down, and a callback is issued to the client with the current, updated, play-list. A callback is also issued with the updated play-list to acknowledge each edit command. Preferably, there is a limit as to how close to air-time a clip normally may be deleted or new material inserted, in order to ensure continuity of transmission of the video stream of each clip. To allow live break-ins or other "emergency" operations, however, the session may be paused and later resumed and subsequent clips may be "trimmed" to reduce their play times to recover the time lost to the break-in.

BSPR:
Although VCR-like functionality is sufficient for streaming data from video files, it is cumbersome to use the VCR functions in a broadcast environment where a stream needs to run continuously with old clips being deleted from the stream as their playback completes and new clips being appended to the stream as their play time approaches. There must be down-time while an existing play-list is being closed and a new one created.

BSPR:
The present invention provides a client-server protocol and interface for providing broadcast playback functionality. In particular, the protocol and interface easily provides continuous play over multiple clips for extended periods of time, allows a play-list to be edited after being given to the video server and during playback of clips in the play-list, allows some notion of "current time" to be used during the streaming of continuous media data, and supports features of the "Louth Automation" video disk communications protocol. The protocol and interface permits an external application to create and manipulate a dynamic play-list; as clips finish playing they are discarded from the play-list and new clips may be appended to or inserted into the play-list at selected positions (subject to server-imposed constraints).

BSPR:
In a preferred mode of operation, the client application first creates a session with a play-list containing a fixed number of entries; the number should be as small as possible consistent with the functions that the client wishes to perform. The client edits this play-list by appending the first few clips and then starts the session playing. Each time transmission of video data of the clip is completed, the clip is removed from the head of the play-list and all other clips are moved down. A callback is issued to the client with the current, updated, play-list. At any time while the video session is playing, edit commands may be issued to insert or delete new continuous media into or from the play-list of the video session. Preferably, there is a limit as to how close to air-time a clip normally may be deleted or new material inserted, in order to ensure continuity of transmission of the video stream of each clip. To allow live break-ins or other "emergency" operations, however, the session may be paused and later resumed and subsequent clips may be "trimmed" to reduce their play times to recover the time lost to the break-in.

DRPR:
FIG. 25 is a block diagram of a clip directory and information associated with each clip including a list of stripe sets comprising the clip;

DRPR:
FIG. 26 is a diagram showing a free stripe set list;

DRPR:
FIG. 27 is a block diagram of a client directory and information associated with each active client including a play list of clips;

DRPR:
FIG. 33 is a flow chart of a program executed by an active controller server when reaching the end of a play-list during the playing of continuous media data for a client;

DEPR:
Turning now to FIG. 10, there is shown a flowchart of a prefetch task including steps for scheduling the transmission of video prefetch commands from one of the stream servers (21 in FIG. 2) to the cache disk array (23 in FIG. 2). As indicated for a first step 101, the video prefetch commands are used when the object being accessed by the stream server is a movie. If so, then in step 102 the stream server finds the next segment for the movie. The media server controller, for example, accesses a movie directory to obtain a list of the addresses of the movie segments in the cached disk array and the size or length of each segment, and transmits this list to the stream server as the object to be accessed. In step 102, the stream server obtains from this list the next segment address and the size of the next segment. Then in step 103 the stream server compares the size of this segment to a predetermined number N which is a limit on the amount of data to be prefetched in response to a single video prefetch command. If the segment size is greater than the number N, then in step 104 only a beginning portion of size N of this segment is prefetched by issuing a video prefetch command to the cached disk array (23 in FIG. 2); the rest of this segment is prefetched in one or more subsequent iterations beginning again in step 103. Otherwise, in step 105, the entire segment is prefetched by issuing a video prefetch command to the cached disk array (23 in FIG. 2). After steps 104 or 105, in step 106 execution branches to step 107 if the end portion of the segment has not been prefetched. In step 107 the segment size is reduced by N, in effect truncating the prefetched portion of the segment. After step 107, the prefetch task is suspended until it is time for the next video prefetch command (issued in steps 104 or 105), and then execution loops back to step 103 to continue prefetching the remaining portion of the segment. Otherwise, at the end of the segment, in step 109 the prefetching task is ended if there are no more segments of the movie to prefetch. If there are more segments of the movie to prefetch, in step 110, the prefetch task is suspended until it is time to prefetch the next segment.

DEPR:
If the track is found to be in the cache in step 122, or after the track is staged into the cache from disk in step 124, then in step 125 the requesting process is placed on a wait list for the track. In this fashion, the track can be retained in the cache until it is fetched by the process. In step 126 a time stamp for the track could also be reset to the current time, and used by a background process in the cached disk array to determine whether any track has been retained in the cache for any inordinate amount of time due to a failure of the process to fetch the video data from the cache. Upon finding that a track has been retained in the cache for an inordinate amount of time, the background process would return the cache slot to the head of the replacement queue and report to the video server manager that the process or processes on the wait list have experienced an error.

DEPR:
Turning now to FIG. 12, there is shown a flowchart of a video fetch routine executed by a channel director (43 in FIG. 3) of the cached disk array in response to a video fetch command from a stream server. In a first step 131, the channel director identifies the next track in the video segment to be fetched. Then in step 132, the channel director accesses the directory in the cache memory (41 in FIG. 3) to determine whether data of the track is in the cache and to determine the cache slot containing the data of the track. If the track is not in the cache, then presumably an error has occurred, because each video fetch command specifying a video segment should have been preceded by a video prefetch command specifying the same video segment, and the video prefetch command should have been executed prior to receipt of the video fetch command. Otherwise, in step 133, the data of the track are transferred from the cache slot to a channel director buffer. Next, in step 134, the data are transferred from the channel director buffer to the stream server having issued the fetch command, and in step 135, the process of the stream server having issued the fetch command is removed from the wait list for the cache slot.

DEPR:
In step 136, execution branches depending on whether the wait list is empty. If so, then in step 137, the cache slot is inserted at the head of the replacement queue, so that the cache slot can be used for receiving data staged from another track. After step 137, or when the wait list is not empty, execution continues to step 138. In step 138, execution loops back to step 131 if there are any more tracks in the segment to be fetched. If not, the video fetch routine is done, and execution returns.

DEPR:
As shown in FIG. 25, a clip directory 281 associates a clip name or identifier 282 with a list 283 of allocated stripe sets, and other information 284 about the clip such as its size in blocks and bytes, ownership, and locking information. The clip directory 281, for example, is organized as a conventional hash table of pointers to associated lists of respective pointers to the information about clips.

DEPR:
The stripe set list associated with each clip, for example, includes a doubly-linked list of entries, and each entry includes a starting stripe set number, an ending stripe set number, and a value indicating the number of data blocks included in the terminal stripe set. Therefore, each entry in the list represents in sequence data blocks beginning in the initial stripe set, continuing in any intermediate stripe set, and ending in the terminal stripe set, and including in the terminal stripe set only the indicated number of data blocks. The stripe set list for each clip can therefore easily be edited by linking and unlinking entries.

DEPR:
Stripe sets are allocated by removing them from a free stripe set list, and de-allocated by returning them to the free stripe set list. As shown

in FIG. 26, for example, each entry in the stripe set free list 291 includes a starting stripe set number and an ending stripe set number, to indicate a range of unallocated stripe sets.

DEPR:
As shown in FIG. 27, the video file server maintains an active client list 301 in order to manage the servicing of client requests. The active client list 301 is essentially a directory to blocks of information about maintaining respective isochronous data streams to the active clients. Such a block of information 302 includes a client identifier 303 identifying the client to which the block of information is relevant, a stream server identifier 304 identifying a stream server assigned to service the client, a play list 305 of clips that are transmitted in sequence to the client, and a current play position 306 in the play list and in the clip currently being played. The play list 305, for example, is a doubly-linked list including, at the head of the list, the clip identifier of the clip currently being transmitted to the client. The video file server responds to a client request for scheduling additional clips by inserting corresponding clip identifiers to the tail of the play list. The video file server responds to a client request to edit its schedule of clips by linking or unlinking corresponding clip identifiers to or from the client's play list.

DEPR:
FIG. 28 shows a preferred placement of the data structures of FIGS. 22 and 25 to 27 in a video file server having the architecture of FIG. 2. As shown in FIG. 28, the free stripe set list 291, the client directory and play lists 300, and the clip directory and stripe set lists 280, are stored in the controller server 28. When a stream server 21 is assigned to service a client 54, the controller server 28 transmits to the stream server a copy of the stripe set list 321 for the client's current clip, and the stripe set list for the client's next clip.

DEPR:
Initially, the video file server is in the idle state 401 for the client. In the idle state 401, an "open play" command is a valid client requests. In response to an "open play" command, the video file server will attempt to allocate resources within the server to enable playing of a specified list of clips. If this attempt is successful, a stream server and a network channel will be assigned to stream continuous media data to the client. To identify the stream, the video file server assigns a unique "handle" to the stream, and returns the handle to the client. Then the continuous media file server enters a "ready for playing" state 406 for the stream. In this state, the stream will be "paused" waiting to start the playing operation at the beginning of the first clip in the specified list of clips to play over the stream.

DEPR:
In the playing state 407 for a client's stream, the video file server sends a callback to the client whenever the playing operation is disrupted, and the callback indicates the nature of the disruption. For example, the callback may indicate that playing has been disrupted by a disk input/output error, or a network input/output error. In the playing state 407 for a client's stream, the video file server also sends a callback to the client when playing of the play list has been completed.

DEPR:
The "rewind" command repositions the current play position of the stream to the start of the first clip of the play list.

DEPR:
The "seek" command positions the current play position of the stream to a position specified by seek arguments of the command. The seek arguments, for example, may specify relative or absolute position in time or in 512 byte blocks in the stream or in the portion of the stream from a specified clip included in the play list for the stream.

DEPR:

Turning now to FIG. 33, there is shown a flow chart of a program executed by the active controller server when reaching the end of a play-list during the playing of continuous media data for a client. When the end of the play-list occurs, as tested in step 381, execution continues to step 382. In step 382, the active controller server returns a callback to the client indicating that the play-list is finished. Next, in step 383, the active controller checks whether a "loop" flag has been set for the stream. In the "open play" command, the client has the option of setting the "loop" flag for selecting whether or not the play list should restart from the beginning when playing of the play list has been completed. If the loop flag has been set, then execution branches from step 383 to step 384 where the play-list pointer is reset to point to the head of the play-list (i.e., a "rewind" operation), and the playing of continuous media data from the play-list continues.

DEPR:
Although the VCR-like functionality of CMFAP described above is sufficient for streaming data from continuous media files, it is cumbersome to use the VCR functions in a broadcast environment where a stream needs to run continuously with old clips being deleted from the stream as their playback completes and new clips being appended to the stream as their play time approaches. There must be down-time while an existing play-list is being closed and a new one created.

DEPR:
To solve these problems, CMFAP has been extended to process a set of commands from the client for providing broadcast playback functionality. In particular, the extended protocol and interface easily provides continuous play over multiple clips for extended periods of time, allows a play-list to be edited after being given to the video server and during playback of clips in the play-list, allows some notion of "current time" to be used during the streaming of continuous media data, and supports features of the "Louth Automation" video disk communications protocol. The extended protocol and interface permits an external application to create and manipulate a dynamic play-list; as clips finish playing they are discarded from the play-list and new clips may be appended to or inserted into the play-list at arbitrary points (subject to server imposed constraints).

DEPR:
A specific example of a format for the play-list is:

DEPR:
Turning now to FIGS. 34 to 35, there is shown a flow diagram illustrating use of the CMFAP broadcast playback commands in a protocol between a client and the video file server. The client first creates a session with a play-list containing a fixed number of entries; the number should be as small as possible consistent with the functions that the client wishes to perform. The client application does this by first sending a "create session" command to the video file server, as show in step 421 of FIG. 34. In response, in step 422 of FIG. 34, the video file server allocates server resources for a broadcast video session to the client's specified destination. The server initially creates the play-list as empty, and it must be populated with at least one clip before playing of a broadcast session may be started. The server returns a "session handle" to the client, to identify the broadcast video session. A format for such a "create session" command is:

DEPR:
Next, in step 423 of FIG. 34, the client receives the session handle, and uses it to send one or more "edit session" commands to the video file server to add one or more clips to the play-list. Each such edit command may manipulate the state of a single clip within the play-list by adding a new clip or deleting an existing clip within the play-list. A format for such a play-list edit command is:

DEPR:
VAPPeditop.sub.-- t is an enumerated type which defines edit operations on a play-list:

DEPR:
In response to each "edit session" command, in step 424, the video file server adds a clip to the play-list, and returns to the client the new version of the play-list.

DEPR:
In response to the resume session command, in step 426, the server begins play of the clips in the play-list, by transmitting continuous media data from the first clip in the play-list to the client's specified destination. Each time transmission of continuous media data of the clip at the head of the play-list is completed, the clip is removed from the head of the play-list, and transmission of continuous media data from the next clip in the play-list is begun immediately, without any interruption in the stream of continuous media data to the client's specified destination. In addition, a callback is issued to the client with the current, updated, play-list. A specific format for such a callback is:

DEPR:
At any time while the video session is playing, edit commands may be issued to delete or insert new material from or into the play list of the video session. For example, as shown in steps 427 and 428 of FIG. 35, the client may respond to the callback from the server when transmission from a clip is completed by sending one or more "edit session" commands to the server to add additional clips to the play-list. The client may also send "edit session" commands to the video file server during playback of the session to remove clips from the play-list. The video file server responds in step 429 by dynamically revising the play-list during broadcast of the clip at the head of the play-list. Preferably, there is a limit as to how close to broadcast time a clip normally may be deleted or new material inserted, in order to ensure continuity of transmission of the continuous media stream of each clip.

DEPR:
It is the client's responsibility to ensure that the play-list does not become empty. As tested in step 430, if the play-list becomes empty for more than a certain timeout period, such as thirty seconds, then in step 431 the server automatically destroys the session. Steps 430 and 431 can be programmed in a fashion similar to the programming shown in FIG. 33 for playing under VCR controls. The server will also destroy the session in response to a "destroy session" command from the client.

DEPR:
As seen in the state diagram of FIG. 36, edit session commands can be issued whenever the session exists; i.e., in the session created state, the session playing state, and the session paused state. Edit commands delete or insert new material from or into the play-list during the session playing state without causing an interruption of the broadcast transmission. To ensure continuity of broadcast transmission during each clip, however, it is desirable to set a limit as to how close to air-time a clip normally may be deleted or new material inserted. If this limit would not be met, the edit command is rejected. To allow live break-ins or other "emergency" operations, however, the session may be paused and later resumed and subsequent clips may be "trimmed" to reduce their play times to recover the time lost to the break-in.

DEPR:
In step 453, the controller server checks whether it is in the "session playing" state for the session identified by the "edit session" command. If not, the possibility of interrupting broadcast transmission does not exist, and execution branches to step 454 to edit the play-list for the session. In step 455, the controller server transmits to the client the edited play-list to acknowledge completion of the "edit session" command, and processing of the "edit session" command is finished.

DEPR:
If the controller server finds in step 453 that the session is playing,

then execution continues to step 456 to check whether the requested edit is too close to air time. In step 456, the controller server computes the difference in time (.DELTA.T) between the earliest time of continuous media to be added to or deleted from the play-list, and the current time. Then, in step 457, the controller server checks whether this difference in time (.DELTA.T) is less than a certain minimum time (TMIN). If not, then execution branches from step 457 to step 454 to edit the play-list. Otherwise, execution continues to step 458. In step 458, the controller server returns to the client an error code indicating that the edit is too close to broadcast time to avoid an interruption, and processing of the "edit session" command is finished.

DEPR:
In view of the above, there has been described a client-server protocol and interface for providing broadcast playback functionality. The protocol and interface easily provides continuous play over multiple clips for extended periods of time, allows a play-list to be edited after being given to the video server and during playback of clips in the play-list, allows some notion of "current time" to be used during the streaming of continuous media data, and supports features of the "Louth Automation" video disk communications protocol. The protocol and interface permits an external application to create and manipulate a dynamic play-list; as clips finish playing they are discarded from the play-list and new clips may be appended to or inserted into the play-list at selected positions (subject to server-imposed constraints).

DEPV:
clips--list of clips to play

DEPV:
info--pointer to a list of clip entries:

DEPV:
count--size of the play-list in clips

DEPV:
status--status reply; e.g., successful, session in wrong state, insufficient bandwidth, internal communications failure, requested clip missing, requested clip empty, bad endpoint, invalid session handle, invalid clip handle, unsupported operation, insufficient internal resources, bandwidth of requested clip is inconsistent with bandwidth requested when the play-list was created, disk I/O error, network I/O error, generic failure, clip already in use for incompatible operation, attempt to edit too late

DEPV:
status--operation status; may indicate that an attempt to edit a clip within a play-list was made too late to maintain continuous playback

DEPV:
play-list--current play-list

DEPV:
VAPPeditDelete--delete a clip from a play-list

DEPV:
VAPPeditAppend--append a new clip to a play-list

DEPV:
isEmpty--true if play-list is now empty

DEPV:
play-list--the current play-list

CLPR:
3. The method as claimed in claim 1, wherein the second command is substantially identical in format to the play-list edit commands.

CLPR:
5. The method as claimed in claim 1, wherein the server returns to the client an acknowledgement of each play-list edit command, and the acknowledgement of each play-list edit command includes an edited version of the play-list resulting from the server editing the play-list in accordance with said each play-list edit command.

CLPR:
6. The method as claimed in claim 1, wherein the server transmits to the client an indication that transmission from a clip is complete each time the server has completed transmission of continuous media data from a clip in the play-list during the broadcast session.

CLPR:
7. The method as claimed in claim 6, wherein said indication that transmission from a clip is complete includes a current version of the play-list.

CLPR:
8. The method as claimed in claim 1, wherein the server transmits to the client an indication that the play-list is empty when transmission of continuous media data from a last clip in the play-list is finished and the play-list becomes empty.

CLPR:
9. The method as claimed in claim 1, wherein the server de-allocates the server resources when the play-list is empty for more than a certain amount of time.

CLPR:
10. The method as claimed in claim 1, wherein the server checks whether or not each of said play-list edit commands specifies a change of continuous media too close to broadcast time to avoid an interruption of transmission of the continuous media data from the server to said destination, and when one of said play-list edit commands is found to specify a change of continuous media too close to broadcast time to avoid an interruption of transmission of the continuous media data from the server to said destination, not editing the play-list in response to said one of said play-list edit commands, and instead returning an error message to the client.

CLPR:
12. The method as claimed in claim 11, wherein the server returns to the client an edited version of the play-list when the server edits the play-list in response to each of the play-list edit commands.

CLPR:
13. The method as claimed in claim 11, wherein the server transmits to the client an indication that transmission from a clip is complete each time the server has completed transmission of continuous media data from a clip in the play-list during the broadcast session.

CLPR:
14. The method as claimed in claim 13, wherein said indication that transmission from a clip is complete includes a current version of the play-list.

CLPR:
15. The method as claimed in claim 13, wherein the server transmits to the client an indication that the play-list is empty when transmission of continuous media data from a last clip in the play-list is finished and the play-list becomes empty.

CLPR:
17. The method as claimed in claim 16, wherein the server transmits to the client an indication that transmission from a clip is complete each time the server has completed transmission of continuous media data from a clip in the play-list during the broadcast session.

CLPR:
18. The method as claimed in claim 17, wherein said indication that transmission from a clip is complete includes a current version of the play-list.

CLPR:
19. The method as claimed in claim 16, wherein the server transmits to the client an indication that the play-list is empty when transmission of continuous media data from a last clip in the play-list is finished and the play-list becomes empty, and wherein the server de-allocates the server resources when the play-list is empty for more than a certain amount of time.

CLPR:
21. The method as claimed in claim 16, wherein the server checks whether or not each of said play-list edit commands specifies a change of continuous media too close to broadcast time to avoid an interruption of transmission of the continuous media data from the server to said destination, and when one of said play-list edit commands is found to specify a change of continuous media too close to broadcast time to avoid an interruption of transmission of the continuous media data from the server to said destination, not editing the play-list in response to said one of said play-list edit commands, and instead returning an error message to the client.

CLPR:
24. The continuous media server as claimed in claim 22, wherein the second command is substantially identical in format to the play-list edit commands.

CLPR:
25. The continuous media server as claimed in claim 24, wherein the controller is programmed for receiving from the client a third command for initiating the broadcast session, and in response beginning the broadcast session by transmitting continuous media data from the first clip in the play-list to said destination.

CLPR:
26. The continuous media server as claimed in claim 22, wherein the controller is programmed for returning to the client an acknowledgement of each play-list edit command, and the acknowledgement of each play-list edit command includes an edited version of the play-list resulting from the server editing the play-list in accordance with said each play-list edit command.

CLPR:
27. The continuous media server as claimed in claim 22, wherein the controller is programmed for transmitting to the client an indication that transmission from a clip is complete each time the server has completed transmission of continuous media data from a clip in the play-list during the broadcast session.

CLPR:
28. The continuous media server as claimed in claim 27, wherein the controller is programmed for including a current version of the play-list in said indication that transmission from a clip is complete.

CLPR:
29. The continuous media server as claimed in claim 22, wherein the controller is programmed for transmitting to the client an indication that the play-list is empty when transmission of continuous media data from a last clip in the play-list is finished and the play-list becomes empty.

CLPR:
30. The continuous media server as claimed in claim 22, wherein the controller is programmed for de-allocating the server resources when the play-list is empty for more than a certain amount of time.

CLPR:
31. The continuous media server as claimed in claim 22, wherein the controller is programmed for checking whether or not each of said play-list edit commands specifies a change of continuous media too close to broadcast time to avoid an interruption of transmission of the continuous media data from the server to said destination, and when one of said play-list edit commands is found to specify a change of continuous media too close to broadcast time to avoid an interruption of transmission of the continuous media data from the server to said destination, for not editing the play-list in response to said one of said play-list edit commands, and instead returning an error message to the client.

CLPR:
33. The continuous media server as claimed in claim 32, wherein the controller is programmed for returning to the client an edited version of the play-list when the controller edits the play-list in response to each of the play-list edit commands.

CLPR:
34. The continuous media server as claimed in claim 32, wherein the controller is programmed for transmitting to the client an indication that transmission from a clip is complete each time the server has completed transmission of continuous media data from a clip in the play-list during the broadcast session.

CLPR:
35. The continuous media server as claimed in claim 34, wherein the controller is programmed for including a current version of the play-list in said indication that transmission from a clip is complete.

CLPR:
36. The continuous media server as claimed in claim 32, wherein the controller is programmed for transmitting to the client an indication that the play-list is empty when transmission of continuous media data from a last clip in the play-list is finished and the play-list becomes empty.

CLPR:
38. The continuous media server as claimed in claim 37, wherein the controller is programmed for transmitting to the client an indication that transmission from a clip is complete each time the server has completed transmission of continuous media data from a clip in the play-list during the broadcast session.

CLPR:
39. The continuous media server as claimed in claim 38, wherein the controller is programmed for including a current version of the play-list in the indication that transmission from a clip is complete.

CLPR:
40. The continuous media server as claimed in claim 37, wherein the controller is programmed for transmitting to the client an indication that the play-list is empty when transmission of continuous media data from a last clip in the play-list is finished and the play-list becomes empty, and for de-allocating the server resources when the play-list is empty for more than a certain amount of time.

CLPR:
42. The continuous media server as claimed in claim 37, wherein the controller is programmed for checking whether or not each of said play-list edit commands specifies a change of continuous media too close to broadcast time to avoid an interruption of transmission of the continuous media data from the server to said destination, and when one of said play-list edit commands is found to specify a change of continuous media too close to broadcast time to avoid an interruption of transmission of the continuous media data from the server to said destination, for not editing the play-list in response to said one of said play-list edit commands, and instead returning an error message to

the client.

CLPR:
45. The program storage device as claimed in claim 43, wherein the second command is substantially identical in format to the play-list edit commands.

CLPR:
46. The program storage device as claimed in claim 45, wherein the program is executable by the server for receiving from the client a third command for initiating the broadcast session, and in response beginning the broadcast session by transmitting continuous media data from the first clip in the play-list to said destination.

CLPR:
47. The program storage device as claimed in claim 43, wherein the program is executable by the server for returning to the client an acknowledgement of each play-list edit command, and for including, in the acknowledgement of each play-list edit command, an edited version of the play-list resulting from the server editing the play-list in accordance with said each play-list edit command.

CLPR:
48. The program storage device as claimed in claim 43, wherein the program is executable by the server for transmitting to the client an indication that transmission from a clip is complete each time the server has completed transmission of continuous media data from a clip in the play-list during the broadcast session.

CLPR:
49. The program storage device as claimed in claim 48, wherein the program is executable by the server for including a current version of the play-list in said indication that transmission from a clip is complete.

CLPR:
50. The program storage device as claimed in claim 43, wherein the program is executable by the server for transmitting to the client an indication that the play-list is empty when transmission of continuous media data from a last clip in the play-list is finished and the play-list becomes empty.

CLPR:
51. The program storage device as claimed in claim 43, wherein the program is executable by the server for de-allocating the server resources when the play-list is empty for more than a certain amount of time.

CLPR:
52. The program storage device as claimed in claim 43, wherein the program is executable by the server for checking whether or not each of said play-list edit commands specifies a change of continuous media too close to broadcast time to avoid an interruption of transmission of the continuous media data from the server to said destination, and when one of said play-list edit commands is found to specify a change of continuous media too close to broadcast time to avoid an interruption of transmission of the continuous media data from the server to said destination, for not editing the play-list in response to said one of said play-list edit commands, and instead returning an error message to the client.

CLPR:
54. The program storage device as claimed in claim 53, wherein the program is executable by the server for returning to the client an edited version of the play-list when the controller edits the play-list in response to each of the play-list edit commands.

CLPR:
55. The program storage device as claimed in claim 53, wherein the

program is executable by the server for transmitting to the client an indication that transmission from a clip is complete each time the server has completed transmission of continuous media data from a clip in the play-list during the broadcast session.

CLPR:
56. The program storage device as claimed in claim 55, wherein the program is executable by the server for including a current version of the play-list in said indication that transmission from a clip is complete.

CLPR:
57. The program storage device as claimed in claim 53, wherein the program is executable by the server for transmitting to the client an indication that the play-list is empty when transmission of continuous media data from a last clip in the play-list is finished and the play-list becomes empty.

CLPR:
59. The program storage device as claimed in claim 58, wherein the program is executable by the server for transmitting to the client an indication that transmission from a clip is complete each time the server has completed transmission of continuous media data from a clip in the play-list during the broadcast session.

CLPR:
60. The program storage device as claimed in claim 59, wherein the program is executable by the server for including a current version of the play-list in the indication that transmission from a clip is complete.

CLPR:
61. The program storage device as claimed in claim 58, wherein the program is executable by the server for transmitting to the client an indication that the play-list is empty when transmission of continuous media data from a last clip in the play-list is finished and the play-list becomes empty, and for de-allocating the server resources when the play-list is empty for more than a certain amount of time.

CLPR:
63. The program storage device as claimed in claim 58, wherein the program is executable by the server for checking whether or not each of said play-list edit commands specifies a change of continuous media too close to broadcast time to avoid an interruption of transmission of the continuous media data from the server to said destination, and when one of said play-list edit commands is found to specify a change of continuous media too close to broadcast time to avoid an interruption of transmission of the continuous media data from the server to said destination, for not editing the play-list in response to said one of said play-list edit commands, and instead returning an error message to the client.

CLPV:
(c) the client receiving the acknowledgement and in response transmitting to the server at least a second command specifying a play-list of continuous media clips from which continuous media data are to be transmitted from the server to said destination during the broadcast session; and then

CLPV:
(d) the server receiving said at least a second command and thereafter beginning the broadcast session by transmitting continuous media data from a first clip in the play-list to said destination; and then

CLPV:
(e) the client sending to the server play-list edit commands during the transmission of continuous media data from at least one clip in the play-list for editing the play-list including the addition of at least one additional clip to the play-list, and the server receiving the